[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 9/445

[12] 发明专利申请公开说明书

[21] 申请号 00819275.8

[43] 公开日 2003 年 8 月 20 日

[11] 公开号 CN 1437726A

[22] 申请日 2000.3.2 [21] 申请号 00819275.8
[86] 国际申请 PCT/US00/05378 2000.3.2
[87] 国际公布 WO01/65365 英 2001.9.7
[85] 进入国家阶段日期 2002.9.2
[71] 申请人 凤凰技术有限公司
地址 美国加利福尼亚州
[72] 发明人 S·阿赖

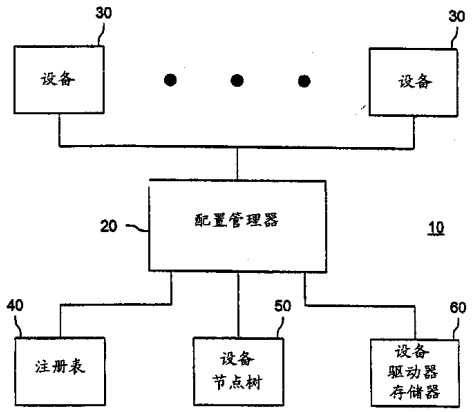
[74] 专利代理机构 中国专利代理(香港)有限公司
代理人 吴立明 王 勇

权利要求书 5 页 说明书 18 页 附图 5 页

[54] 发明名称 在计算机运行中交换设备的装置与方法

[57] 摘要

一种用于交换安装在计算机系统中的设备的系统允许用户在该计算机系统正在运行或在睡眠状态中时进行设备交换。 该系统能识别、访问及使用新增加的设备而无须重新起动或重新引导该计算机系统。



ISSN1008-4274

1. 一种枚举计算机系统中存在的第一设备而使该计算机系统能识别该第一设备的方法，该计算机系统具有存储器及使用存储在该存储器中的数据结构来建立计算机系统的工作配置的操作系统，该数据结构具有对应于该计算机系统中存在的设备的一或多个设备节点，该方法包括下述步骤：

当计算机系统正在操作时向该操作系统提供已将该第一设备插入计算机系统中的指示；

响应该指示枚举用于控制该第一设备的第一设备控制器；

响应第一设备控制器的枚举初始化与该第一设备控制器关联的第一驱动器；

用该初始化后的第一驱动器定位与该第一设备控制器关联的枚举器；以及

用该定位的枚举器枚举该第一设备，

其中当初始化该第一驱动器时，该第一驱动器只能定位与第一设备控制器关联的枚举器。

2. 对照权利要求1的方法，其中枚举第一设备控制器的步骤包含将该第一设备控制器的设备节点加到该数据结构上的子步骤。

3. 按照权利要求1的方法，其中枚举该第一设备控制器的步骤包含从存储在计算机系统的存储器中的设备节点表中定位该第一设备控制器的设备节点的子步骤。

4. 按照权利要求1的方法，其中枚举第一设备的步骤包含从存储在计算机系统的存储器中的设备节点表中定位第一设备的设备节点的子步骤。

5. 按照权利要求4的方法，其中枚举第一设备的步骤还包含将该第一设备的定位的设备节点加到该数据结构上的子步骤。

6. 按照权利要求1的方法，其中枚举第一设备的步骤包含将该第一设备的设备节点加到该数据结构上的子步骤。

7. 按照权利要求1的方法，还包括下述步骤：

在将第一设备连接到计算机系统上之后向该第一设备供电；以及接通在第一设备与第一设备控制器之间传输的任何数据信号。

8. 按照权利要求1的方法，在接通第一设备的步骤之前，还包括

下述步骤:

确定当前是否正在访问计算机系统中的第二设备;

从该数据结构中清除对应于该第二设备的设备节点及对应于用于控制该第二设备的控制器的设备节点;

隔离在第二设备与第二设备控制器之间传输的任何数据信号从第二设备控制器到第二设备的传输;以及

关闭第二设备的供电。

9. 按照权利要求 8 的方法,其中清除对应于第二设备及第二设备控制器的设备节点的步骤包含卸载与该第二设备及该第二设备控制器关联的所有驱动器的子步骤。

10. 按照权利要求 8 的方法,其中该第一设备控制器与第二设备控制器是同一控制器。

11. 按照权利要求 8 的方法,还包括将隔离的数据信号接通到第一设备上的步骤。

12. 按照权利要求 1 的方法,其中该第一设备是软驱动器、硬驱动器、及 CD-ROM 驱动器之一。

13. 按照权利要求 1 的方法,其中该第一设备控制器为 IDE 驱动器控制器及软驱动器控制器之一。

14. 按照权利要求 8 的方法,其中该第二设备为软驱动器、硬驱动器、及 CD-ROM 驱动器之一。

15. 按照权利要求 8 的方法,其中该第二设备控制器为 IDE 驱动器控制器及软驱动器控制器之一。

16. 一种计算机程序产品,包括:

具有用于枚举计算机系统中存在的第一设备而使该计算机系统能识别该第一设备的计算机可读的程序代码模块实现在其上的计算机可用的介质,该计算机系统具有存储器及使用存储在该存储器中的数据结构来建立计算机系统的工作配置的操作系统,该数据结构具有对应于该计算机系统中存在的设备的一或多个设备节点,该计算机程序产品包括:

第一枚举模块,用于在该计算机操作中将第一设备插入该计算机系统中之后枚举用于控制该第一设备的第一设备控制器;

初始化模块,用于响应该第一设备控制器的枚举初始化与该第一

设备控制器关联的第一驱动器;

定位器模块，用于用初始化的第一驱动器识别与该第一设备控制器关联的枚举器；以及

第二枚举模块，用于用与该第一设备控制器关联的该定位的枚举器枚举该第一设备，

其中在初始化第一驱动器时该第一驱动器只能定位与该第一设备控制器关联的枚举器。

17. 按照权利要求 16 的计算机程序产品，其中该第一枚举模块包含用于将第一设备控制器的设备节点加到该数据结构上的子模块。

18. 按照权利要求 16 的计算机程序产品，其中该第一枚举模块包含用于从存储在计算机系统的存储器中的设备节点表中定位该第一设备控制器的设备节点的子模块。

19. 按照权利要求 16 的计算机程序产品，其中该第二枚举模块包含用于从存储在计算机系统的存储器中的设备节点表中定位该第一设备的设备节点的子模块。

20. 按照权利要求 19 的计算机程序产品，其中该第二枚举模块还包含用于将该第一设备的定位的设备节点加到该数据结构上的子模块。

21. 按照权利要求 16 的计算机程序产品，其中该第二枚举模块包含用于将该第一设备的设备节点加到该数据结构上的子模块。

22. 按照权利要求 16 的计算机程序产品，还包括:

用于在将该第一设备连接到该计算机系统上之后向该第一设备供电的供电模块；以及

用于接通在该第一设备与第一设备控制器之间传输的任何数据信号的开关模块。

23. 按照权利要求 16 的计算机程序产品，还包括:

查询模块，用于确定当前是否正在访问计算机系统中的第二设备;

清除模块，用于从该数据结构中消除对应于第二设备的设备节点及对应于控制该第二设备的控制器的设备节点;

隔离模块，用于隔离在第二设备与第二设备控制器之间传输的任

何数据信号从第二设备控制器到第二设备的传输;

供电模块，用于切断第二设备的供电，使得能将该第二设备从计算机系统上清除。

24. 按照权利要求 23 的计算机程序产品，其中该第一设备控制器与第二设备控制器为同一控制器。

25. 按照权利要求 23 的计算机程序产品，还包括用于将隔离的数据信号接通到第一设备上的开关模块。

26. 按照权利要求 23 的计算机程序产品，其中该清除模块包含用于卸载与第二设备及第二设备控制器关联的所有驱动器的子模块。

27. 按照权利要求 16 的计算机程序产品，其中该第一设备为软驱动器、硬驱动器、及 CD-ROM 驱动器之一。

28. 按照权利要求 16 的计算机程序产品，其中该第一设备控制器为 IDE 驱动器控制器及软驱动器控制器之一。

29. 按照权利要求 23 的计算机程序产品，其中该第二设备为软驱动器、硬驱动器、及 CD-ROM 驱动器之一。

30. 按照权利要求 23 的计算机程序产品，其中该第二设备控制器为 IDE 驱动器控制器及软驱动器控制器之一。

31. 一种用于枚举计算机系统中存在的第一设备而使该计算机系统能识别该第一设备的计算机系统，该计算机系统具有存储器及使用存储在该存储器中的数据结构来建立该计算机系统的工作配置的操作系统，该数据结构具有对应于该计算机系统中存在的设备的一或多个设备节点，该计算机系统包括:

在计算机系统操作时向操作系统提供已将第一设备插入该计算机系统中的指示的第一部件;

响应该指示枚举用于控制该第一设备的第一设备控制器的第二部件;

响应第一设备控制器的枚举，枚举与该第一设备控制器关联的第一驱动器的第三部件;

用初始化的第一驱动器定位与该第一设备控制器关联的枚举器的第四部件; 以及

用该定位的枚举器枚举第一设备的第五部件;

其中在初始化该第一驱动器时，该第一驱动器只能定位与该第一设备控制器关联的枚举器。

在计算机运行中交换设备的装置与方法

发明领域

本发明一般涉及与计算机互联的设备的领域，而更具体地，涉及在计算机操作中交换或改变与计算机互联的设备。

背景技术

完全“即插即用”操作系统的驱动器与设备能以极少到无须用户的主动介入来处理设备的增加与消除。结果完全“即插即用”系统允许相对不老练的用户容易与平顺地安装与交换设备，并且用户无须理解软件或具有编程能力。

在各操作系统内，存在着一系列用于发现与初始化计算机中所包括的设备以便它们能正确地与该计算机的所有资源连通的驱动器。对于诸如 Windows 95®或 98®（微软公司产品）等操作系统，当在操作系统的安装或引导期间起动驱动器时，驱动器枚举或识别连接在计算机上的设备。然而对于一些设备，它们的相关设备驱动器是不能完全“即插即用”的。这些驱动器的实例包含用于硬盘驱动的 IDE 驱动器及用于软盘驱动的软盘驱动器。因为这些驱动器不能完全“即插即用”，这些设备的驱动器在运行时间期间不能枚举设备。

由于这一限制，在系统运行（热交换 hot swapping）或睡眠状态（温交换 warm swapping）时，这些操作系统不能支持诸如硬盘驱动器、CD-ROM 驱动器及软盘驱动器等不具有完全“即插即用”驱动器的设备的交换。而是，用这些设备之一来交换另一个时要求用户重新起动或重新引导计算机。这一要求是对用户既费时又费事的。

发明概述

简要的，本发明包括枚举计算机系统中存在的第一设备而使计算机系统能识别该第一设备的方法，该计算机系统具有存储器及使用存储在该存储器中的数据结构来建立计算机系统的工作配置的操作系统，该数据结构具有对应于计算机系统中存在的设备的一或多个设备节点，该方法包括下述步骤：在计算机系统操作时向操作系统提供第一设备已插入计算机系统中的指示；响应该指示为控制该第一设备枚举一控制器；响应第一设备控制器的枚举，初始化与该第一设备控制

器关联的驱动器；用初始化的驱动器定位与该第一设备控制器关联的枚举器（enumerator）；以及用定位的枚举器枚举该第一设备，其中在初始化时与第一设备控制器关联的驱动器只能定位与该第一设备控制器关联的枚举器。

在本方法的另一方面中，在连接第一设备之前，该方法还包括下述步骤：判定当前是否正在访问计算机系统中的第二设备；从数据结构中清除对应于第二设备的设备节点及对应于用于控制该第二设备的控制器的设备节点；将在第二设备与第二设备控制器之间传输的任何数据信号从第二设备控制器到第二设备的传输中隔离；以及关闭第二设备的供电。

在本方法的又另一方面中，清除对应于第二设备及第二设备控制器的设备节点的步骤包含卸载与第二设备及第二设备控制器关联的所有驱动器的子步骤。

在本发明的又一方面中，计算机程序产品包括具有实现在其中的用于枚举计算机系统中存在的第一设备而使计算机系统能识别该第一设备的计算机可读的程序代码模块的计算机可用的介质，该计算机系统具有存储器及使用存储在该存储器中的数据结构来建立该计算机系统的工作配置的操作系统，该数据结构具有对应于存在在计算机系统中的设备的一或多个设备节点，该计算机程序产品包含：第一枚举模块，用于枚举在计算机系统操作中将第一设备插入计算机系统之后枚举用于控制第一设备的控制器；初始化模块，用于响应第一设备控制器的枚举初始化与第一设备控制器关联的驱动器；定位器模块，用于用初始化的驱动器识别与第一设备控制器关联的枚举器；以及第二枚举模块，用于利用与第一设备控制器关联的定位的枚举器枚举第一设备，其中与第一设备控制器关联的驱动器在初始化时只能定位与第一设备控制器关联的枚举器。

附图说明

图 1 为符合本发明的计算机系统的框图。

图 2a 与 2b 为符合本发明的交换进程的框图流程图。

图 3 为符合本发明的枚举进程的框图流程图。

图 4 为符合本发明的设备交换系统的框图。

发明详述

下面在特定实施例的上下文中描述本发明，但并不旨在这样限定本发明。

图 1 示出如何将符合本发明的计算机系统 10 的部件安装在一起。如图 1 中所示，计算机系统 10 包含配置管理器 20、多个设备或资源 30、登记处（registry）40、设备节点（devnode）树 50、及设备驱动器存储器 60。见下面对上文中元件的定义。最好在即插即用框架中实现计算机系统 10。虽然图 1 中示示出，计算机系统 10 包含其它元件，诸如处理器、存储器、监视器、及象键盘或指点设备等输入设备。如本技术中已知的，其它元件也可包含在计算机系统 10 中。

下面是计算机系统 10 中的这些元件的定义。可以作为虚拟设备驱动器（VxD）实现的“配置管理器”20 是即插即用框架的中心部件。VxD 是专门为 Windows 95 与 98 格式化的驱动器。包含 DLL 与 DOS TSR 格式在内的其它驱动器也是可能有的。配置管理器 20 指导计算机系统 10 中的所有配置管理。为了指导配置管理，它能使用三个数据存储器：注册表 40、设备节点树 50 及设备驱动器存储器 60，并能用四种软件部件工作：枚举器、仲裁器、设备加载器及设备驱动器。

“枚举器”确定计算机系统 10 中当前安装了哪些硬件设备及它们需要哪些资源。从而，枚举便是发现或识别设备。如同配置管理器 20，枚举器也能实现为 VxD。识别出各设备之后，枚举器便建立“设备节点”。设备节点是包含关于识别的设备的基本信息的数据结构。设备节点能包含指示可能的配置、当前配置、状态信息、及该设备的驱动器的字段。

枚举器识别出新设备之后，称作“设备安装器”的模块在“注册表”40 中建立一新的项。注册表 40 存储项的分层树，它们包含曾经安装在计算机系统 10 中的各设备的关键字与值。各项的关键字为各自的设备的标识，而各项中的值包含关于设备的类型的信息及其说明，在枚举时应为特定设备加载哪些驱动器、驱动器修正号、制造商、及设备的潜在逻辑配置。值的信息能来自设备零售商提供的设备信息文件或来自设备本身。“设备信息文件”包含关于已知设备及它们的配置的信息。为了为刚安装在系统中的设备建立设备节点，枚举器能利用

对应于注册表 40 中的设备的项的信息。

将枚举器建立的各设备节点放置在“设备树”50 中。设备树 50 为描述计算机系统 10 中存在的所有硬件的设备节点的分层树。设备树中的层次构成为使得将设备节点放置在控制这些设备节点的设备的设备节点的下方层次中，借此提供控制的指示。例如，将受总线控制的设备的设备节点放置在总线的设备节点下方。诸如总线的设备节点等设备节点称作父节点，而诸如连接在总线上的设备等受父节点控制的设备节点则为子节点。

枚举器识别出设备并建立了其设备节点之后，配置管理器 20 调用与该设备关联的“设备加载器”。能作为 VxD 实现的设备加载器加载与管理给定设备节点的设备驱动器及枚举器。在由好几个层的设备驱动器管理设备时，通常使用设备加载器。设备加载器协调好几个层的工作并进行适用于所有设备驱动器的通用配置任务。

“设备驱动器”控制与管理设备硬件。设备驱动器负责识别给定设备的特定资源需求并设定该设备以配置管理器 20 分配的配置操作。能将已安装在计算机系统 10 中的所有设备 30 的设备驱动器存储在设备驱动器存储器 60 中。

最后，也能作为 VxD 实现的“仲裁器”是由配置管理器 20 用来解决设备 30 之间的资源冲突的。仲裁器审阅安装在计算机系统 10 中的设备 30 的需求表并找出资源的最佳分配来满足所有设备。仲裁器为计算机系统 10 中的若干资源作出无冲突分配，其中包含输入/输出（I/O）地址、存储器地址、中断请求（IRQ）、及直接存储器存取（DMA）通道。

应指出仲裁器、枚举器、注册表、设备节点树、配置管理器及设备加载器是包含微软公司在 Windows 95 与 Windows 98 中提供的在内的各种软件零售商所提供的软件的标准项目。

下面是计算机系统 10 中的配置管理器 20 所使用的数据存储器与软件部件的更详细的说明。建立计算机系统的工作配置的积木块是设备节点。如上所述，设备节点是物理上存在在系统上的设备 30 的基本存储器内表示。它是包含关于设备及其驱动器的信息的数据结构。存储在设备节点中的信息包含设备标识（ID）、其当前分配的配置、可能的逻辑配置、状态信息、及设备驱动器入口点。如上所述，设备节

点是由枚举器建立的。响应设备节点的建立，配置管理器20导致适当驱动器的加载。

用在设备节点数据结构中的“设备ID”为对计算机系统10中可包含的各设备独一无二的ASCII串。该串通常标识找到该设备的部件，但由于它也用作注册表40中的关键字，它必须是在计算机系统10内独一无二的，以便能可靠地检索到关于该设备的信息。这一串能用来交叉参照关于存储在注册表40中的设备的数据。

下面是设备ID的实例：Root*PNP 0000\0；ISAPNP\ADP1522-DEV0000\E8124123；BIOS*PNP0100\0；及PCMCIA\MEGAHERTZ-XJ124FM-936B。如这些实例中所见，ID的第一部分标识找到该设备的部件。例如，ISAPNP对应于即插即用的设备的ISA总线。设备ID的这一命名方案为总线特定的命名方案。

如上面所讨论的，枚举器为识别安装的硬件与建立设备节点的部件。在计算机系统10中，各总线包含由配置管理器20调用来定位在其各自的总线上的设备的枚举器。这些总线包含PCI、EISA、PCMCIA、SCSI等。这些总线上的设备通常配置成完全支持即插即用的。也能将ISA总线配置成支持即插即用的，并称作ISAPNP总线。在总线上识别出的设备的设备节点可具有它们自己的枚举器而在设备节点树50上建立子节点。这些子设备节点的任何一个能具有在设备节点树50上建立该设备节点的进一步子节点的枚举器。

为了识别在不能即插即用的标准ISA总线上配置的设备（有时称作传统设备）配置管理器20能使用检测模块而不是枚举器。检测模块能使用识别传统设备的方法，诸如检验硬编码的I/O位置的预期值来检测诸如键盘或中断控制器等设备。

枚举器负责为它们定位的设备建立设备ID。如上所述，各标识符应是独一无二的并在每一次系统引导时始终如一。应将该串构成为使它能用于标识注册表40中独一无二的项。该串通常以枚举器的名称开始，后面跟随一“\”。例如，ISA即插即用枚举器能将所有设备标识符起始为“ISAPNP\”。

枚举器构筑设备节点树50并可参预在与各自的枚举器关联的总线上配置设备30。在大多数操作系统中，每一次计算机起动时发生枚举进程。在枚举进程结束时，设备节点树50包含关于可利用的系统资源、

所需的设备驱动器、及这些资源的其它要求的信息。

设备节点树 50 为表示计算机系统 10 中的硬件的结构的设备节点的分层的树。从而，设备节点是在设备节点树 50 的根上或者是父设备节点的子节点。一个设备节点能具有多个子设备节点。设备节点的父节点的其它子节点称作兄弟。每一次起动计算机时构造设备节点树 50。此外，它可以是完全动态的，如果在计算机系统 10 中消除或增加设备 30，甚至在计算机正在运行时改变。

设备节点树 50 为计算机系统 10 提供许多功能。首先，它描述计算机系统 10 中存在的所有硬件。此外，设备节点树 50 为枚举器提供配置它们的子硬件的机制，允许独立于总线的驱动器。驱动器能是独立于总线的，因为各总线的枚举器能独立于其它总线识别与配置受该总线控制的设备。通常，驱动器并不知道或关心它们在树内的精确位置。驱动器通常只关心它们自己的硬件设备或设备节点，而不注意树中它们上面或下面的层。

设备节点树 50 包含计算机系统 10 中所有设备 30 的配置信息，并且配置管理器 20 用它来跟踪与各设备关联的资源，诸如中断请求（IRQ）、输入/输出（I/O）端口及甚至聚 SCSI 标识符等非共享资源。设备节点树 50 描述存在的设备与资源、资源需求、资源依存性、及当前资源分配。

表 1 示出说明某些典型设备的设备 ID 及可期望在层次中何处能找到它们的设备节点树的实例。未示出设备 ID 以外的各节点的实际内容。

表 1：设备节点树实例

Htree\Root\0-设备节点树的根节点
\Root*PNP0801\0-老式声音清除器兼容的声音设备
\Root*PNP0C00\0-即插即用 BIOS
\BIOS*PNP0901\0-超级 VGA 兼容的显示器适配器
\BIOS*PNP0000\0-AT 中断控制器
\BIOS*PNP0100\0-AT 定时器
\BIOS*PNP0200\0-AT DMA 控制器
\BIOS*PNP0301\0-PS/2 形式键盘控制器
\BIOS*PNP0400\0-标准 LPT 打印机端口
\BIOS*PNP0400\1-标准 LPT 打印机端口
\BIOS*PNP0501\0-16550 通信端口
\BIOS*PNP0501\1-16550 通信端口
\BIOS*PNP0700\0-PC 标准软盘控制器
\BIOS*PNP0800\0-AT 形式扬声器声音
\BIOS*PNP0901\0-SVGA 兼容的显示器适配器
\BIOS*PNP0B00\0-AT 实时时钟
\BIOS*PNP0C01\0-系统存储器
\BIOS*PNP0E00\0-PCMCIA 控制器
\PCMCIA\3C08SF\0-网络适配器

如表 1 中所示，根节点是声音清除器与即插即用 BIOS 的父节点。即插即用 BIOS 又具有一系列子节点。即插即用 BIOS 的枚举器识别这些设备的每一个并为各设备建立设备节点，其中各设备节点为即插即用 BIOS 设备节点的子节点。如上面讨论的，这一分层结构示出这些子节点的每一个是受即插即用 BIOS 控制的。类似地，PCMCIA 控制器的枚举器识别受该 PCMCIA 控制器控制的网络适配器，并为该网络适配器建立一设备节点作为 PCMCIA 控制器的子节点。

计算机系统 10 还使用与扩充注册表 40 及各种设备信息文件（带.INF 文件名扩展）。注册表 40 存储曾经安装在特定机器上的所有

设备 30，其中包含诸如传统设备等不允许完全即插即用的设备。注册表 40 还存储已由枚举器定位的即插即用设备的有关信息、设备特定的状态信息、设备对设备驱动器的连接、以及关于最后知道的配置的数据-用于加速资源分配。

上面提到的设备信息文件包含关于已知设备及它们的配置的信息。计算机系统 10 能提供某些类别信息文件（例如 SCSI.INF 与 MODEM.INF）；其它是由设备制造商提供的。当枚举器检测到或枚举新设备时，在所有已知的.INF 文件中搜索与该设备 ID 匹配的项。在找到匹配时，便拷贝适当的文件并将所需的数据拷贝到注册表 40 中新建立的注册表条目中。

注册表 40 是通过微软 Win 32® API 组在 Windows 95、Windows 98 与.Windows NT® 中访问的全系统数据库。此外，Windows 95 与 Windows 98 提供对注册表 40 的实方式访问以便在切换到保护方式之前在引导期间能使用它。如前面指出的，注册表 40 存储类似于 Windows 3.1-格式.INI 文件中的项的“关键字”与“值”的分层树（除外 Windows 3.1 .INI 文件具有与注册表的树结构相反的平坦结构）。在枚举特定设备时操作系统利用注册表 40 来存储关于应加载哪些驱动器的信息、以及诸如驱动器修正号、制造商、及该设备的潜在逻辑配置等信息。下面的表 II 示出示范性注册表条目。

表 II: 注册表条目实例

\HKEY_LOCAL_MACHINE\ENUM\ROOT*PNP0000\0000
DrvDesc="可编程中断控制器"
类别="系统"
BootConfig=(I/O 端口 20h，21h，A0h，及 A1h-二进制资源数据)
驱动器="系统\0000"
硬件 ID="*PNP0000"
InfName="MACHINE.INF"
Mfg="微软"
NoSetupUI="1"
ENUM\BIOS*PNP0000\0000\LogConfig
0=(二进制资源数据，在这一情况中作为上面的 BootConfig)
SYSTEM\CurrentControlSet\Services\Class\System\0000
DriverDesc="可编程中断控制器"
DevLoader\"*VPICD"
EnumPropPages="Sysclass.dll,EnumPropPages"

表 II 中的注册表条目示出可为特定设备包含在注册表 40 中的信息的实例。在这一实例中，设备为可编程的中断控制器。这一设备的信息包含可能的引导配置、与该设备关联的设备驱动器、独一无二的 ID、该设备的信息文件名、及用于加载该设备的设备驱动器的设备加载器。

大多数即插即用注册表条目存储在注册表 40 的 ENUM 树中，即 \HKEY_LOCAL_MACHINE\ENUM。在 ENUM 下，各枚举器具有其本身的分支，它曾经枚举与建立的各设备具有在该枚举器下面的一个子节点。存在着一个称作 ENUM\ROOT 的特殊枚举器分支，它包含诸如传统设备等非即插即用的硬件的注册表条目，对于它们没有枚举器但能用上面讨论的其它手段检测到。例如，在安装操作系

统时，根据诸如 MSDET.INF 信息文件等信息文件中的表加载检测模块。当检测到不完全允许即插即用的设备时，便从注册表 40 中的适当设备信息文件中检索关于它的信息。然后将这一信息存储在注册表 40 中建立在 HKEY_LOCAL_MACHINE\ENUM\Root 分支下的新节点中。

当枚举器或检测模块检测到在注册表 40 中没有它的注册表条目的设备时，设备安装器在注册表 40 中建立新项。大多数信息是由设置有该设备的.INF 文件或直接由该设备的建立模块提供的。注册表 40 还存储配置特定的信息，它是与是否与有能力的便携式计算机对接或脱离对接有关的信息，以及用户特定的信息。

设备节点树 50 与注册表 40 之间的区别是重要的。在完全即插即用的系统中，设备节点树 50 是当前安装的设备的精确分层表示。反之，注册表 40 包含曾经安装（即使它们当前不存在）的所有设备的信息。设备在注册表 40 中的位置并不完全反映设备节点树 50 中描述的分层结构。在注册表 40 中，设备位于它们的枚举器下方，但所有枚举器是保持在平坦的表中的，意味着枚举器并不按照识别它们的相关设备的枚举器保持在分层结构中。设备节点树 50 只存在在存储器中，而注册表 40 则保存在盘上。

如前面指出的，设备驱动器控制与管理设备硬件。设备驱动器负责识别给定设备 30 的特定资源需求并将该设备设定为以配置管理器 20 所赋予的配置操作。配置管理器 20 提供设备驱动器能用来执行其任务的服务与消息。

设备驱动器检验设备节点树 50 中表示其设备的分配的配置的设备节点部分来确定已为该设备分配了哪些资源。对于即插即用卡，给定设备的分配的资源在每一次引导时，或甚至在计算机系统 10 正在运行时（动态地）可以改变。逻辑配置描述对给定的硬件有效的各种配置。诸如它是否断开或配置等关于设备的状态的信息是由配置管理器 20 维护在设备节点树 50 中的各设备节点中的。驱动器能通过向配置管理器 20 查询该信息来查询这一信息。在为特定的设备节点加载驱动器时，它向配置管理器 20 注册，传递在分配或改变硬件配置时将调用的设备的入口点。虽然设备驱动器或其它部件不能直接访问设备节点树 50 或设备节点，它能检索关于设备节点的本体的信息并执行检索及设置该

设备节点中的信息的任务。配置管理器20提供这些服务在设备节点上执行任务。然而，设备驱动器及其它部件能在注册表40中存储关于给定的设备节点的信息，以及检索配置管理器20存储在注册表40中的关于设备节点的信息。

设备加载器加载与管理给定设备节点的设备驱动器及枚举器。当设备由设备驱动器的不同的层管理时，通常使用设备加载器。设备加载器协调不同的层的工作并执行适用于所有设备驱动器的通用配置任务。

配置管理器20能将设备加载器与给定的设备节点关联。设备加载器也能为给定的设备节点或任何子设备节点加载枚举器及设备驱动器。其它服务包含检索与分配私有值给设备节点以及在设备描述符块中检索与设备节点关联的任何动态加载的虚拟设备。

仲裁器为分配诸如IRQ与I/O端口等各种资源的所有权的部件。资源仲裁器调解设备之间的资源冲突。仲裁器检验设备需求表并找出最佳资源分配来满足所有设备。配置管理器20也提供服务与信息来支持资源仲裁器的操作。

需求表标识设备30成功地操作所需的资源类型及与这些资源关联的任何限制。IRQ、I/O端口、DMA通道与存储器范围是资源类型的实例。限制通常是资源间的依存性，诸如设备30要求必须见到组合在一起的IRQ3与I/O端口O2F8以便成功地操作。

逻辑配置是设备30成功地操作所要求的资源的说明。任何给定的设备30可具有若干可能的逻辑配置。配置管理器20利用设备30的逻辑配置来决定如何在竞争中的设备之间分配资源。配置管理器20提供设备驱动器、枚举器、及其它部件能用来检验与建立逻辑配置的服务。

资源描述符描述逻辑配置中的资源。其中有存储器、I/O端口、DMA通道、及IRQ的资源描述符。能根据需要建立其它资源描述符来标识设备可利用的其它类型的资源。

存储器资源描述符标识存储器地址范围。这一描述符包含描述存储器资源的MEM_DES结构及标识各设备的可能存储器配置的MEM_RANGE结构的数组。I/O端口资源描述符标识I/O端口地址范围。它包含描述I/O端口资源的IO_DES结构及标识可能的端口配置

的 IO_RANGE 结构的数组。

DMA 通道资源描述符标识一组 DMA 通道选择对象。这一描述符为标识设备能使用的 DMA 通道的 DMA_DES 结构。最后，IRQ 资源描述符标识一组 IRQ 选择对象。它是标识设备能使用的 IRQ 及 IRQ 是否能共享的 IRQ_DES 结构。

当以 I/O 端口资源描述符工作时，IOR_Alias 与 IOR_Decode 值规定设备对其应答的端口别名。端口别名是设备 30 对之应答的地址，似乎它便是 I/O 端口的实际地址。此外，某些卡实际使用附加端口于不同目的，但使用使它们似乎正在使用别名的解码方案；例如，ISA 卡可解码 10 位并需要 03Coh。这一卡必须规定 04h 的 IOR_Alias 偏移及 IOR_Decode 3（不使用别名作为实际端口）。为了方便，可将别名字段设定为零来指示不需要别名；在这一情况中，忽略解码字段。

如果该卡要使用 7 Coh、OBCoh、及 OFCoh 上的端口，其中这些端口具有不同的功能，IOR_Alias 值将是相同的而 IOR_Decode 值将是 OFh，指示端口地址的位 11 与 12 是有效的。从而，分配是对所有端口 (PORT[i]+(n*alias*256))ϕ(decode*256/03FFh)，其中 n 为任何整数而 PORT 为 IOR_nPorts、IOR_Min、及 IOR_Max 字段规定的范围。注意最小别名是 4 而最小解码是 3。

由于 ISA 总线的历史原因，假设使用其中 PORT=n*400+Z（其中“Z”为范围 100h-3ffh 内的端口，而“N”大于或等于 1）的任何端口的所有 EISA 与 ISA 卡将保留端口 Z 并将其它端口作为别名对待。如果保留了落入这一组中的端口但驱动器并不保留“Z”地址，则假设该设备是在局部总线上（诸如 PCI），其中该保留的端口地址不出现在 ISA 总线上。

范围表是 I/O 端口或存储器地址范围（以 DWORD 形式）的排序的表，其中没有两个范围重叠。管理 I/O 与存储器资源的资源仲裁器利用该范围表服务来发现给定的 I/O 或存储器范围是否与任何其它范围冲突。这些服务检测建立包含重叠的范围表的企图并或者失败或者在已经存在重叠时建立合并的范围。各范围必须指定连接的地址范围，但范围表本身可包含多个范围，其中没有一个必须与表中的任何其它范围连接。

与这些部件及资源联合工作，配置管理器 20 为计算机系统 10 的所有设备 30 找出可工作的配置从而各设备 30 能与其它设备无冲突地使用其分配的 IRQ 号、I/O 端口地址、及其它资源。配置管理器 30 还协助监视计算机系统 10 中存在的设备 30 的数目与类型的改变，并在发生改变时根据需要管理设备 30 的配置。为了建立与维护配置，配置管理器 20 与枚举器、资源仲裁器、设备加载器、及设备驱动器联合工作。它提供这些部件用来执行它们的配置任务的服务与信息。下面的表 III 示出配置管理器 20 所提供的一些服务的表及这些服务的各个的说明。

表 III: 配置管理器服务

服务	说明
CONFIGMG_Create_DevNode	在硬件树上增加设备节点。
CONFIGMG_Disable_DevNode	禁止硬件树中的设备节点。
CONFIGMG_Enable_DevNode	允许硬件树中的设备节点。
CONFIGMG_Query_Remove_SubTree	检验是否能消除设备节点及其后代。
CONFIGMG_Reenumerate_DevNode	使指定的设备节点的枚举器接受 CONFIG_ENUMERATE 枚举器功能。
CONFIGMG_Remove_SubTree	消除设备节点及其子节点。

可将这些服务用在计算机系统 10 中来允许在计算机系统 10 正在操作时在计算机系统 10 中插入与识别设备 30。除了表 III 中所示的服务，还有许多在 Windows 95 与 98 中标准的服务能用在计算机系统 10 中。Karen Hazzah 的“编写 Windows VxD 及设备驱动器，第二版”(1997 年 R&D Books 出版）提供即插即用框架及其部件的更完整说明，通过引用将其结合在此。

图 2a 与 2b 为符合本发明的用于在操作中的计算机系统 10 中消除

与插入具有不完全允许“即插即用”的驱动器的设备的进程的流程图。这一进程能作为软件程序、布线成执行该进程的诸如可编程存储器、固件或处理器等硬件设备或它们的某种组合实现。参见图 2a，在引导计算机系统 10 时，配置管理器 20 用枚举器、设备节点、设备驱动器及注册表 40 生成设备节点树 50（步骤 110）。

在计算机系统 10 正在运行（热交换）或在睡眠状态中（温交换）时，计算机系统 10 的用户能请求用当前存在在计算机系统 10 中的设备替换或交换不同的设备（步骤 112）。为了作出这一请求，用户能通过 Windows 型界面或与计算机系统 10 正在使用的操作系统兼容的任何其它类型界面查询计算机系统 10。在这一请求中，用户能指定要消除的设备及要插入的新设备。此外，如果没有设备要被消除，则用户可以只指定要插入的设备。响应这一请求，配置管理器 20 调用用户请求消除的设备的控制器的驱动器（步骤 114）。这一调用能用 CONFIGMG_Query_Remove_SubTree 服务作出。如果要消除的设备是硬驱动器，这一设备控制器可以是例如 IDE 控制器，而如果要消除的设备是软驱动器，则为软控制器。取决于消除的设备的类型，其它控制器也是可能的。

设备控制器的驱动器确定要消除的设备当前是否正在使用（步骤 116）。例如，如果该设备是盘驱动器，驱动器会确定该驱动器上的文件当前是否打开。为了作出确定，可作出 CONFIGMG_Query_Remove_SubTree 调用或查询。在盘驱动器的实例中，可通过若干层驱动器访问该盘驱动器，诸如文件系统驱动器、盘驱动器驱动器及盘控制器驱动器。虽然盘控制器驱动器可能不知道盘驱动器上打开的文件，它能将关于状态的查询提交给能作出确定的其它驱动器层。

如果该设备当前正在使用，通过界面通知用户不能消除该设备（步骤 118）。如果在应答该查询中返回 CR_REMOVED_VETOED 指示，则确定该设备当前正在使用。在盘驱动器的实例中。可以向用户提供信息来关闭该文件并在关闭文件之后重试交换。反之，如果该设备不在使用，则通过界面通知用户能进行交换（步骤 120）。如果返回的指示为 CR_SUCCESS 则该设备不在使用。

在消除该设备之前，配置管理器 20 从设备节点树 50 中消除与要

消除的设备控制器关联的所有设备节点及受该设备控制器控制的所有设备，其中包含要消除的该设备（步骤 122）。这能用 CONFIGMG_Remove_SubTree 服务完成。作为这一操作的后果，从计算机系统 10 中卸载了与消除的设备节点相关的所有驱动器与驱动器字母。例如，消除软盘驱动器及其控制器的设备节点会消除它们的相关驱动器及驱动器字母，诸如“a”驱动器。

如果正在交换的设备是受同一控制器控制的，诸如都受 IDE 控制器控制的 CD-ROM 与硬驱动器，则只消除该控制器的设备节点。然而，如果正在交换的设备是受不同控制器控制的，则有必要消除两个控制器的设备节点。如果正在插入新设备而不消除另一设备，则只须消除控制该新设备的控制器的设备节点。

消除了设备节点之后，便断开了要消除的设备与其控制器之间传输的信号（步骤 124）。为了执行这一信号断开，可将一电路集成到控制器中或加到计算机系统 10 中控制器与要消除的设备之间。这一信号断开电路可以是诸如开关设备，它在断开时封锁控制器与要消除的相关设备之间的信号传输。信号断开电路的开关控制响应能交换该设备的指示。虽然信号断开并非必要，但最好包含以免对控制器及要消除的设备的可能损坏。如果要交换的设备是受不同控制器控制的，也必须断开要插入的设备的控制器的信号。如果要插入的设备不是交换另一设备的，断开要插入的设备的控制器的信号也是必要的。

除了断开设备与其控制器之间传输的信号之外，切断正在消除的设备的供电（步骤 126）。一旦信号已断开及该设备已断电，用户便能从计算机系统 10 中消除该设备（步骤 128）。

如图 2b 中所示，在用户从计算机系统 10 中消除该设备之后，用户便能插入新设备（步骤 130）。在作出这一交换之前，可通过界面向用户提供计算机系统 10 已准备好进行交换的指示。

在向新设备供电与重新接通断开的信号之前，用户能通过界面提供已插入新设备的指示。作为替代，可在计算机系统 10 中包含传感器电路来感测或确定已插入新设备，并提供已插入新设备的指示。

给出了已插入新设备的指示，便向新设备供电（步骤 132）并将从上述控制器断开的信号重新接通到该新设备（步骤 134）。响应这一指

示，配置管理器 20 随即指令枚举器枚举或识别该新增加的设备的控制器（步骤 136）。这一枚举器与该新设备的控制器连接在其上面的总线通信。枚举可用 CONFIGMG_Reenumernt_DevNode 系统调用完成。通过重新枚举该控制器，定位新设备的控制器并将其设备节点加到设备节点树 50 上（步骤 138）。如果交换的设备是受不同的控制器控制的，则也重新枚举消除的设备的控制器。

在将控制器的设备节点加到设备节点树 50 上之后，还初始化对应于该控制器的驱动器（步骤 140）。如上面所讨论的，控制器的设备节点可包含关于该控制器的驱动器的位置的信息。在驱动器的初始化期间，该控制器的驱动器定位与该控制器关联的枚举器，后者将该新设备枚举为连接在该控制器上（步骤 142）。作为这一枚举的结果，也将对应于该新设备的设备节点加到设备节点树 50 上（步骤 144）。设备节点树 50 中存在了新设备的设备节点，计算机系统 10 便能识别与使用该新设备，似乎该新设备是在引导时安装的一样。

如上面所讨论的，当在计算机系统 10 正在运行时插入设备时，诸如 Windows 95 与 98 等操作系统不能识别与初始化具有不允许完全“即插即用”的驱动器的设备。反之，这些驱动器只能在初始化了对应的驱动器时才能枚举设备，而这在传统上只在起动或安装操作系统期间出现。然而在图 2a 与 2b 的进程中，控制器与设备的驱动器是卸载然后重新初始化的，这便允许甚至在计算机系统 10 正在运行或睡眠时也能识别与初始化新安装的设备。

识别计算机系统 10 中存在的设备的进程依赖于枚举器的使用。枚举器能与特定的设备节点关联，诸如总线或设备控制器的设备节点。此外，至少一个枚举器与设备节点树 50 的根节点关联。配置管理器 20 使用与设备节点树 50 的根节点关联的枚举器在起动时开始该枚举进程。然而，不管枚举器是与根节点、父节点还是子节点关联的，枚举进程都以相同方式进行。图 3 示出符合本发明的枚举进程的流程图。

首先，配置管理器 20 或与设备节点关联的设备驱动器判定设备节点是否包含关联的枚举器（步骤 210）。如果否，则该设备节点不能具有任何子节点与关联的设备来枚举并为其调用驱动器。反之，如果配置管理器 20 或设备驱动器判定该设备节点包含枚举器，便调

用该枚举器（步骤 212）。然后调用的枚举器搜索与对应的设备节点的设备关联或受其控制的任何设备（步骤 214）。如果定位了这一设备，枚举器便在注册表 40 中搜索对应于该定位的设备的项（步骤 216）。

如果在注册表 40 中未找到对应于定位的设备的项，称作设备安装器的模块为该定位的设备在注册表 40 中建立一新项（步骤 218）。一旦为该设备建立了项，便以与任何其它安装的或以前安装的设备相同的方式对待它。如果找到或已新建立了项，枚举器便将该项的设备节点加到设备节点树 50 上（步骤 220）。加到设备节点树 50 上的定位的设备的设备节点是作为对具有找到该定位的设备的枚举器的设备节点的子节点加入的。例如，如果 IDE 控制器的设备节点的枚举器定位了硬驱动器，该硬驱动器的设备节点将作为设备节点树 50 中的 IDE 控制器的设备节点的子节点加入。枚举器还根据设备节点中的信息识别该定位的设备的设备驱动器（步骤 222），并使用这一信息来调用该设备驱动器（步骤 224）。然后该进程重复自己。

图 4 示出符合本发明的设备交换器 70 的框图。设备交换器 70 可作为软件程序、诸如可编程存储器、固件或处理器等硬件设备、或者它们的某种组合实现。设备交换器 70 可作为诸如图 1 中所示的计算机系统 10 等计算机系统的一部分加入。例如，如果作为软件程序实现，设备交换器能在激活时加载到计算机系统 10 的存储器中。

如图 4 中所示，设备交换器 70 包含查询模块 72、消除模块 74、断开与供电模块 76 及枚举模块 78。这些模块能在计算机系统 10 操作时加载到计算机系统 10 的存储器中。操作中，图 4 中的计算机系统 10 的用户能用输入设备作出请求用安装在计算机系统 10 中的设备 30 交换另一设备。设备交换器 70 的查询模块 72 接受这一请求，它判定要消除的设备当前是否正受访问（图 2a 的步骤 116）。如果是，查询模块 72 通过诸如计算机系统 10 中的监视器等界面通知用户该设备当前正在使用而此时不能消除（图 2a 的步骤 118）。如果不在访问，查询模块 72 通知用户可以接受交换（图 2a 的步骤 120）。

消除该设备之前，消除模块 74 从设备节点树 50 中消除该设备的设备节点及其相关控制器（图 2a 的步骤 122）。这一消除同时卸载与该设备及控制器相关的所有驱动器及驱动器的驱动器字母。除了消除

设备节点，断开与供电模块76信令断开电路（未示出）断开在被消除的设备与控制器之间传输的信号（图2a的步骤124）。断开与供电模块76还切断被消除的设备的供电（图2a的步骤126）。

交换设备之后，断开与供电模块76接收已插入新增加的设备的指示，向新增加的设备供电并信令断开电路重新接通控制器与增加的设备之间传输的信号（图2b的步骤132与134）。也接收已插入新增加的设备的指示的枚举模块78便定位新增加的设备的控制器并将该控制器的设备节点加到设备节点树50中（图2b的步骤136与138）。这一控制器可以是与消除的设备的控制器同一或不同的控制器。初始化与该控制器的设备节点关联的设备驱动器并调用与该控制器关联的枚举器，后者定位新增加的设备并将其设备节点加到设备节点树50上（图2b的步骤140、142与144）。此时，计算机系统10的其余部分完全可以访问该新增加的设备。

已为了示例与说明的目的提出了本发明的较佳实施例的上文中的描述。这不是为了将本发明穷尽或限制在公开的精确方式上，而是根据上面的教导或者从本发明的实践中有可能进行各种修改与改型。这一实施例是为了说明本发明的原理及使熟悉本技术的人员能在各种实施例中利用本发明及适合于设想的特殊用途的各种修改的实际应用而选择与描述的。旨在由这里所附的权利要求及它们的等效物来定义本发明的范围。

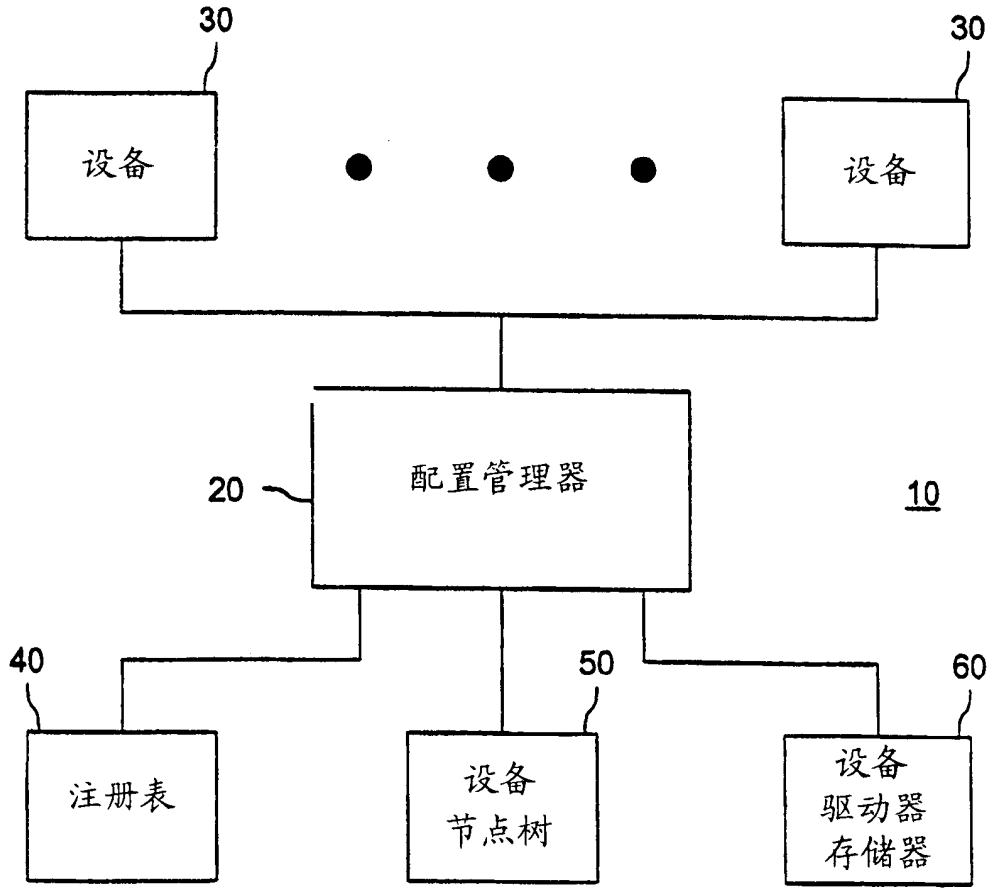


图 1

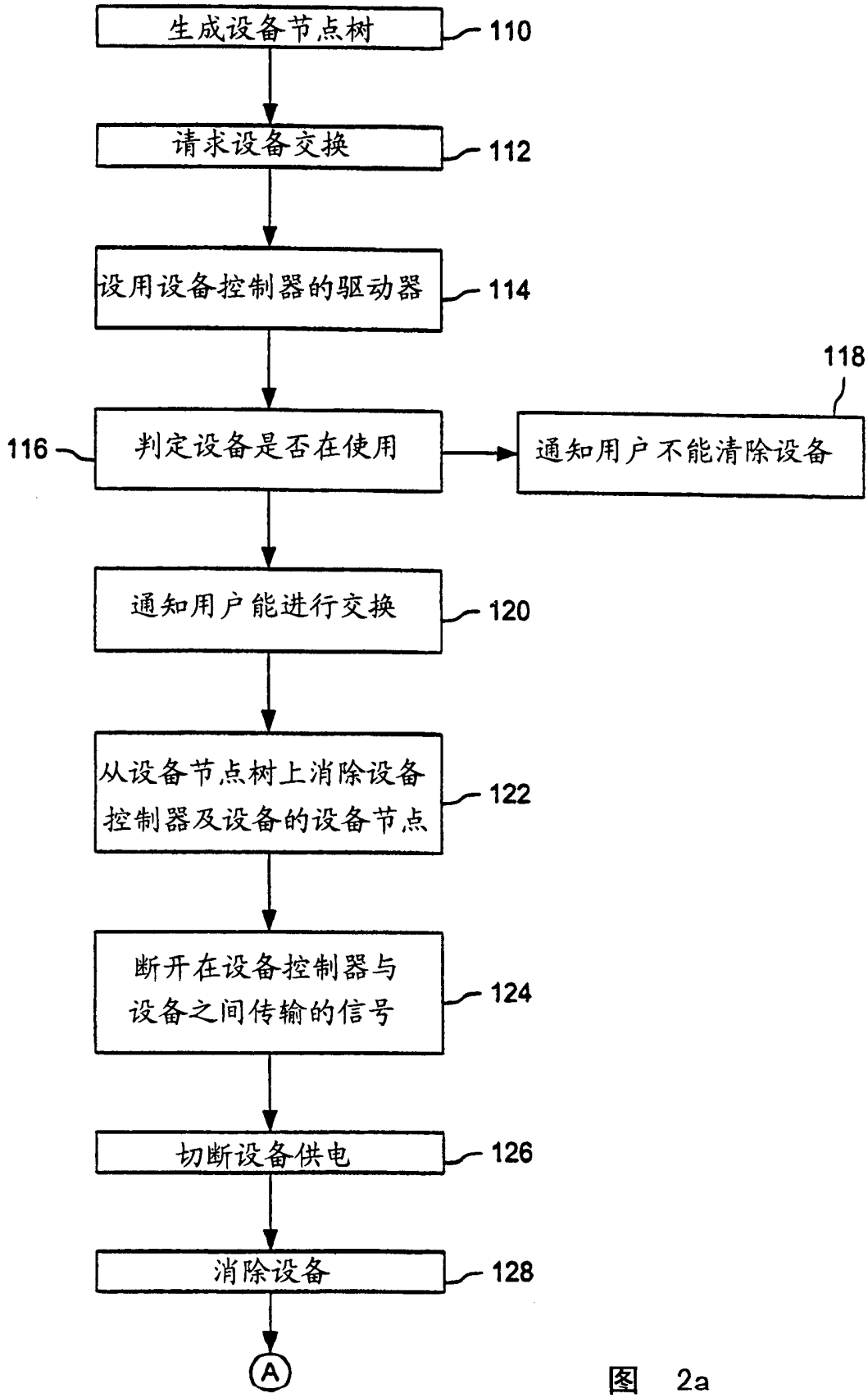

生成设备节点树
110
请求设备交换
112
设用设备控制器的驱动器
114
118
116
判定设备是否在使用
通知用户不能清除设备
通知用户能进行交换
120
从设备节点树上消除设备控制器及设备的设备节点
122
断开在设备控制器与设备之间传输的信号
124
切断设备供电
126
消除设备
128
A

图 2a

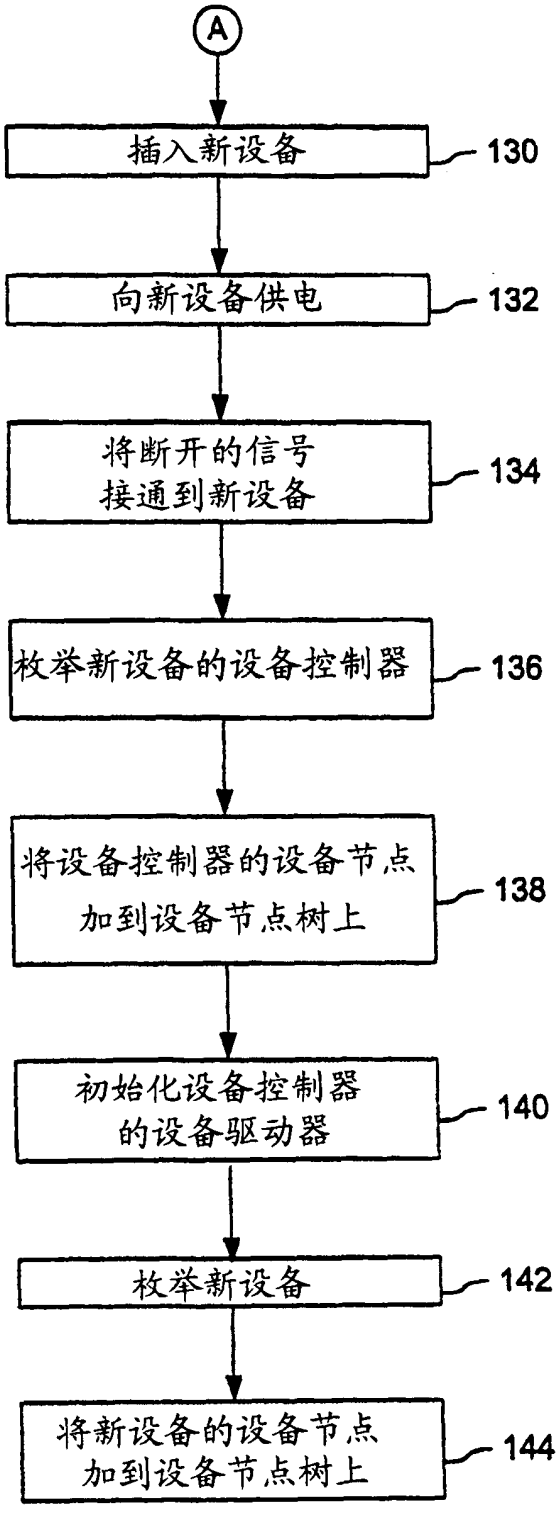

A
插入新设备
130
向新设备供电
132
将断开的信号
接通到新设备
134
枚举新设备的设备控制器
136
将设备控制器的设备节点
加到设备节点树上
138
初始化设备控制器
的设备驱动器
140
枚举新设备
142
将新设备的设备节点
加到设备节点树上
144

图 2b

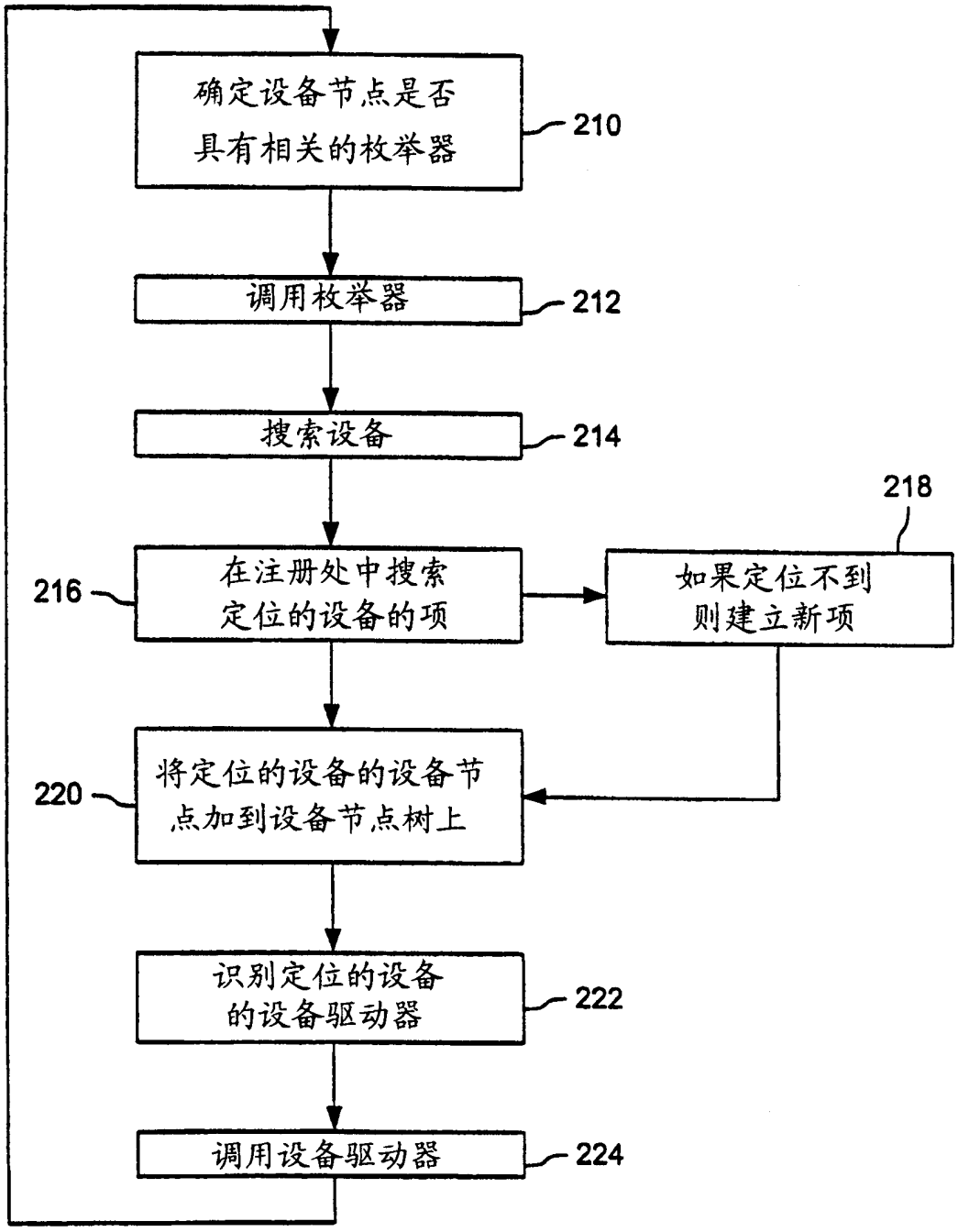

确定设备节点是否
具有相关的枚举器
210
调用枚举器
212
搜索设备
214
218
在注册处中搜索
定位的设备的项
216
如果定位不到
则建立新项
将定位的设备的设备节
点加到设备节点树上
220
识别定位的设备
的设备驱动器
222
调用设备驱动器
224

图 3

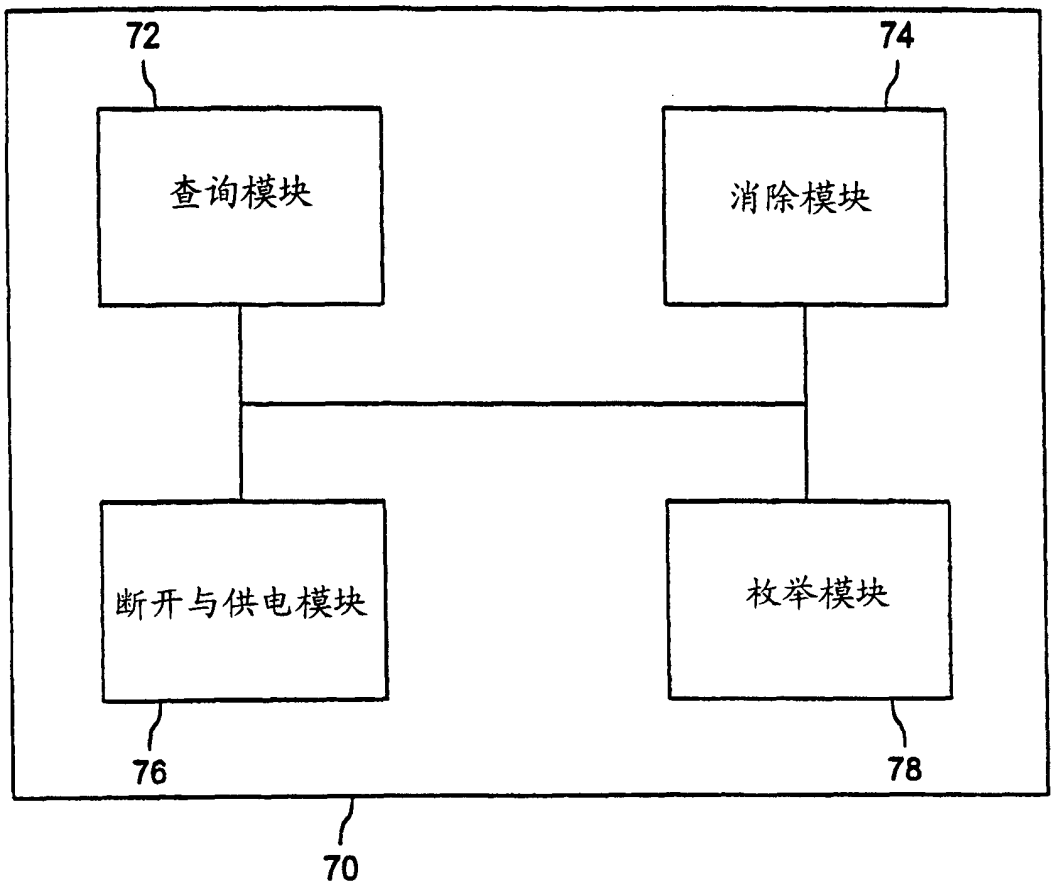


图 4